Systems
Optimization
Laboratory

THE IMPLEMENTATION OF A LAGRANGIAN-BASED ALGORITHM FOR SPARSE NONLINEAR CONSTRAINTS[†]

by

B.A. Murtagh[‡] and M.A. Saunders

TECHNICAL REPORT SOL 80-1
January 1980

Department of Operations Research
Stanford University
Stanford, CA 94305

SYSTEMS OPTIMIZATION LABORATORY
DEPARTMENT OF OPERATIONS RESEARCH
Stanford University
Stanford, California
94305

THE IMPLEMENTATION OF A LAGRANGIAN-BASED ALGORITHM FOR SPARSE NONLINEAR CONSTRAINTS[†]

by

B.A. Murtagh[‡] and M.A. Saunders

TECHNICAL REPORT SOL 80-1
January 1980

[†]Presented at the Tenth International Symposium on Mathematical Programming, Montreal, August 27-31, 1979.

[‡]The University of New South Wales, Sydney, Australia.

Research and reproduction of this report were partially supported by the Department of Energy Contract DE-AS03-76-SF00326, PA Number DE-AT-03-76ER72018; the National Science Foundation Grants MCS76-20019 A01 and ENG77-06761; the U.S. Army Research Office Contract DAAG29-79-C-0110, and the Applied Mathematics Division, DSIR, New Zealand.

CONTENTS

CONTENTS Cont.

ABSTRACT

An algorithm is described for solving large-scale nonlinear programs whose objective and constraint functions are smooth and continuously differentiable. The algorithm is of the augmented Lagrangian type, involving a sequence of sparse, linearly constrained subproblems whose objective functions include a modified Lagrangian term and a modified penalty function.

The algorithm has been implemented in a general purpose Fortran code called MINOS/AUGMENTED. The system is intended for use on problems whose Jacobian matrix is sparse. (Such problems usually include a large set of purely linear constraints.) The bulk of the data may be assembled using a standard linear-programming matrix generator. Function and gradient values for all nonlinear terms are supplied by two user-written subroutines.

Some aspects of the implementation are described in detail, and computational results are given for some nontrivial test problems. Assuming convergence occurs, the work involved is comparable to the solution of a moderate number of linear programs of similar size.

1. INTRODUCTION

The work reported here was prompted by consideration of various ways to extend the linearly constrained optimization code MINOS (Murtagh and Saunders [13]) to include the capability of solving nonlinearly constrained problems. In particular we are concerned with large, sparse problems, in the sense that each variable is associated with relatively few constraints.

Ignoring sparsity for the moment, consider the model problem

$$\begin{array}{ll} \text{minimize} & f^0(\underline{x}) \\ \text{subject to} & \underline{f}(\underline{x}) = \underline{0}, \quad \underline{\ell} \leq \underline{x} \leq \underline{u} \end{array} \tag{1.1}$$

where the functions of $\underline{x}$ are assumed to be twice differentiable with bounded Hessians. For this problem the algorithm discussed here would solve a sequence of linearly constrained subproblems of the form

$$\min \; L(\underline{x},\underline{x}_k,\underline{\lambda}_k,\rho) = f^0(\underline{x}) - \underline{\lambda}_k^T(\underline{f} - \tilde{\underline{f}}) + \frac{1}{2}\rho\,(\underline{f} - \tilde{\underline{f}})^T(\underline{f} - \tilde{\underline{f}}) \tag{1.2a}$$

$$\text{s.t.} \quad \tilde{\underline{f}} = \underline{0}, \qquad \underline{\ell} \leq \underline{x} \leq \underline{u} \tag{1.2b}$$

where $\tilde{\underline{f}}$ is a linear approximation to $\underline{f}(\underline{x})$ at some point $\underline{x}_k$. (Thus $\tilde{\underline{f}} = \underline{f}_k + J_k(\underline{x} - \underline{x}_k)$ where $\underline{f}_k$ and J_k are the constraint vector and Jacobian matrix evaluated at $\underline{x}_k$.) With $\rho = 0$, subproblem (1.2) corresponds to that used by Robinson [20]. The same subproblem (with $\rho = 0$) is used in Phase 2 of Rosen's algorithm [21].

The expression (1.2a) will be called a modified augmented Lagrangian. When $\underline{x}_k$ and $\underline{\lambda}_k$ are taken to be the solution and corresponding Lagrange multipliers for the previous subproblem, Robinson has shown for the case

$\rho = 0$ that the sequence $\{(\underline{x}_k, \underline{\lambda}_k)\}$ will converge quadratically to a solution of the original problem (1.1) as long as the initial pair $(\underline{x}_0, \underline{\lambda}_0)$ is sufficiently close to that solution. A case for which convergence can be expected is when the modified Lagrangian $L(\underline{x}, \underline{x}_k, \underline{\lambda}_k, 0)$ is convex. Since this is not always true, the penalty term involving ρ is included here to ensure that the Hessian of $L(\underline{x}, \underline{x}_k, \underline{\lambda}_k, \rho)$ is positive definite within an appropriate subspace. It also inhibits large discrepancies between $\underline{f}$ and $\underline{\tilde{f}}$, thereby discouraging large changes in $\underline{x}$ in each subproblem if the nonlinearities are such that the linearized constraints have little meaning far from the point of linearization. As always, the intention is to allow convergence from a wider range of starting points. Use of (1.2) represents an alternative to Phase 1 of Rosen's algorithm [21] in which (1.2a) is replaced by $f^0(\underline{x}) + \frac{1}{2}\rho\underline{f}^T\underline{f}$ and the linearized constraints $\underline{\tilde{f}} = \underline{0}$ are deleted from (1.2b).

The reason for choosing the modified penalty in (1.2a) rather than the conventional $\frac{1}{2}\rho\underline{f}^T\underline{f}$ will become clear when sparsity is reintroduced.

1.1. Subproblems

It has been argued in the past that the need to solve linearly constrained subproblems is a drawback of methods such as Robinson's. However when projection (or reduced-gradient or variable-reduction) methods are used we would take the view that linearly constrained subproblems are actually easier to solve than the unconstrained subproblems encountered in other Lagrangian- and penalty-based methods. (Certainly the implementation is more complex but with linear constraints present the optimization usually takes place in a subspace of much smaller dimension.)

For a certain class of objective functions, the development of MINOS has opened the way to solving large linearly constrained problems quite efficiently. Hence for large versions of problem (1.1) involving a sparse Jacobian matrix and many purely linear constraints, it is natural to apply MINOS to the corresponding subproblems (1.2). The resulting extension of MINOS is called MINOS/AUGMENTED and is documented in [14]. Our aim is to describe the algorithm used and some details of its practical implementation, and to discuss its performance on some nontrivial problems.

Note that the Lagrangian and penalty terms in (1.2a) require continual evaluation of the nonlinear constraint functions during the solution of (1.2). In some cases this may be expensive. MINOS/AUGMENTED therefore allows the option of setting $\underline{\lambda}_k = \underline{0}$ and $\rho = 0$ so that only $f^0(\underline{x})$ remains in (1.2a). Some results obtained using this option are also reported.

The MINOS code for linearly constrained optimization is briefly described in Section 2, and Section 3 discusses the method for handling nonlinear constraints. Details of the computer implementation are described in Section 4, and Sections 5 and 6 present some test problems and a discussion of their solution.

2. BRIEF DESCRIPTION OF MINOS

MINOS solves problems expressed in the following standard form:

$$\text{minimize} \quad \underline{f}(\underline{x}) + \underline{c}^T\underline{x} + \underline{d}^T\underline{y} \qquad (2.1)$$

$$\text{subject to} \quad A\begin{bmatrix} \underline{x} \\ \underline{y} \end{bmatrix} = \underline{b} \qquad (2.2)$$

$$\underline{\ell} \le \begin{bmatrix} \underline{x} \\ \underline{y} \end{bmatrix} \le \underline{u} \qquad (2.3)$$

where A is m by n, $m \le n$, and the variables are partitioned into "nonlinear" and "linear" variables $\underline{x}$ and $\underline{y}$ respectively. (This standard form is a slight generalization of the one normally used for linear programming problems; it emphasizes the fact that nonlinearities in the objective function often involve just a few of the variables $\underline{x}$.)

For numerous practical reasons the last m columns of A form the identity matrix I, and the last m components of $\underline{y}$ are the usual logical ("slack" or "surplus") variables.

MINOS uses an "active constraint" strategy, with the general constraints and some portion of the bound constraints being active at any given time. Thus if A is partitioned as $[B \;\; S \;\; N]$ where N is a set of "nonbasic" columns, the active constraints are always of the form

$$\begin{bmatrix} B & S & N \\ & & I \end{bmatrix} \begin{bmatrix} \underline{x}_B \\ \underline{x}_S \\ \underline{x}_N \end{bmatrix} = \begin{bmatrix} \underline{b} \\ \underline{b}_N \end{bmatrix} . \qquad (2.4)$$

The first part of this equation is equivalent to

$$A \begin{bmatrix} \underline{x} \\ \underline{y} \end{bmatrix} = \underline{b} ,$$

while the second part reading $\underline{x}_N = \underline{b}_N$ indicates that the nonbasic variables $\underline{x}_N$ are being held equal to one or other of their bounds. (The components of $\underline{b}_N$ come from $\underline{\ell}$ or $\underline{u}$ as appropriate and the partition $[\underline{x}_B, \underline{x}_S, \underline{x}_N]$ is some permutation of $[\underline{x}, \underline{y}]$.)

The remaining columns of A are partitioned into "basic" and "superbasic" sets B and S, such that the basis matrix B is square and nonsingular. The corresponding basic and superbasic variables $\underline{x}_B$ and $\underline{x}_S$ are free to vary between their bounds during the next iteration.

It can readily be shown that an optimal solution of the above form exists for which the number of superbasic variables is less than or equal to the number of nonlinear variables.

2.1. Some Aspects of the Algorithm Used in MINOS

The operators

$$\hat{A} = \begin{bmatrix} B & S & N \\ & & I \end{bmatrix}, \qquad Z = \begin{bmatrix} -B^{-1}S \\ I \\ 0 \end{bmatrix} \tag{2.5}$$

will be useful for descriptive purposes. The active constraints (2.4) are of the form $\hat{A}\hat{\underline{x}} = \hat{\underline{b}}$, and Z happens to satisfy $\hat{A}Z = 0$.

Under suitable conditions a feasible descent direction $\underline{p}$ may be obtained from the equations

$$Z^TGZ\underline{p}_S = -Z^T\underline{g}\ , \qquad \underline{p} = Z\underline{p}_S \tag{2.6}$$

(see [6]). In particular, if the reduced gradient $Z^T\underline{g}$ is nonzero, and if the reduced Hessian Z^TGZ is positive definite (or if any positive definite matrix is used in place of Z^TGZ), then the point $\underline{x} + \alpha\underline{p}$ lies on the active constraints and some scalar $\alpha > 0$ exists for which the objective function has a lower value than at the point $\underline{x}$.

Other matrices Z exist satisfying $AZ = 0$, but the form chosen above, together with a sparse LU factorization of B, allows efficient computation of the products $Z^T\underline{g}$ and $Z\underline{p}_S$. (Neither B^{-1} nor Z is computed.) A positive-definite approximation to Z^TGZ is maintained in the form R^TR where R is upper triangular. Quasi-Newton updates to R lead to superlinear convergence.

Let the gradient of the objective function (2.1) be the vector $\underline{g} = (\underline{g}_B\ \underline{g}_S\ \underline{g}_N)^T$. If $\underline{\pi}$ satisfies

$$B^T\underline{\pi} = \underline{g}_B \quad . \tag{2.7}$$

it is easily seen that the reduced gradient is

$$Z^T\underline{g} = \underline{g}_S - S^T\underline{\pi} \quad . \tag{2.8}$$

Hence in linear programming terminology the reduced gradient is obtained by "pricing" the superbasic columns S. This is a cheap operation once π has been computed.

Likewise for $\underline{p}$ we have $R^TR\underline{p}_S = -Z^T\underline{g}$ and then

$$\underline{p} = \begin{bmatrix} \underline{p}_B \\ \underline{p}_S \\ \underline{p}_N \end{bmatrix} = Z\underline{p}_S = \begin{bmatrix} -B^{-1}S\underline{p}_S \\ \underline{p}_S \\ \underline{0} \end{bmatrix}, \tag{2.9}$$

so most of the work lies in solving $B\underline{p}_B = -S\underline{p}_S$. (The value $\underline{p}_N = \underline{0}$ indicates that no change will be made to the current nonbasic variables. As long as the reduced gradient $Z^T\underline{g}$ is nonzero, only the variables in $[B\ \ S]$

are optimized. If any such variables encounter an upper or lower bound they are moved into N and the partition [B S] is suitably redefined.)

Note that if the reduced gradient does prove to be zero $(Z^T\underline{g} = \underline{0})$ the reduced objective has reached its optimal value. If we compute $\underline{\sigma} = \underline{g}_N - N^T\underline{\pi}$ (i.e. the usual pricing of nonbasic columns) we then have

$$\begin{bmatrix} B^T & \\ S^T & \\ N^T & I \end{bmatrix} \begin{bmatrix} \underline{\pi} \\ \underline{\sigma} \end{bmatrix} = \begin{bmatrix} \underline{g}_B \\ \underline{g}_S \\ \underline{g}_N \end{bmatrix} \tag{2.10}$$

so that $\underline{\pi}$ and $\underline{\sigma}$ are exact Lagrange multipliers for the current active constraints. The components of $\underline{\sigma}$ indicate whether any nonbasic variables should be released from their bounds. If so, one or more are moved from N into S and optimization continues for the new set [B S]. If not, an optimum has been obtained for the original problem.

In practice, optimization for each [B S] will be curtailed when $Z^T\underline{g}$ is sufficiently small, rather than zero. In this case $\underline{\pi}$ will be just an approximation to the Lagrange multipliers for the general constraints. The accuracy of $\underline{\pi}$ will depend on the size of $||Z^T\underline{g}||$ and on the condition number of the current basis B.

2.2. Key Points

The algorithm implemented in MINOS provides a natural extension of linear programming technology to problems whose objective function is nonlinear. If the number of nonlinear variables is moderate (or more precisely, if the number of superbasic variables and hence the dimension of R is moderate) then the work per iteration is not substantially greater than for one iteration of the revised simplex method on the same data

$$A \begin{bmatrix} \underline{x} \\ \underline{y} \end{bmatrix} = \underline{b}, \qquad \underline{\ell} \leq \begin{bmatrix} \underline{x} \\ \underline{y} \end{bmatrix} \leq \underline{u} \ .$$

Here we assume that the cost of evaluating the objective function and its gradient is moderate compared to manipulation of a sparse factorization of the basis matrix B. At the same time it is important that the step-size α be determined efficiently. The line-search procedure used in MINOS is that of Gill, Murray _et al_. ([7]), which allows the user to control the accuracy of the search by means of a parameter ETA, where $0.0 \leq \text{ETA} < 1.0$. Even with a relatively accurate search (e.g. ETA = 0.01) the number of function and gradient evaluations required is typically very few (e.g. 1, 2 or 3 per search). This is increasingly beneficial for the algorithm discussed next, where the objective function is modified to include an arbitrary number of nonlinear functions.

3. EXTENSION TO NONLINEAR CONSTRAINTS

3.1. Statement of the Problem

It is assumed that the nonlinearly constrained problem can be expressed in the following standard form:

$$\text{minimize} \quad f^0(\underline{x}) + \underline{c}^T\underline{x} + \underline{d}^T\underline{y} \tag{3.1}$$

$$\text{subject to} \quad \underline{f}(\underline{x}) + A_1\underline{y} = \underline{b}_1 \qquad (m_1 \text{ rows}) \tag{3.2}$$

$$A_2\underline{x} + A_3\underline{y} = \underline{b}_2 \qquad (m_2 \text{ rows}) \tag{3.3}$$

$$\underline{\ell} \leq \begin{bmatrix} \underline{x} \\ \underline{y} \end{bmatrix} \leq \underline{u} \qquad m = m_1 + m_2 \tag{3.4}$$

where $\underline{f}(\underline{x}) = [f^1(\underline{x}), \ldots, f^{m_1}(\underline{x})]^T$. The first n_1 variables $\underline{x}$ are again called "nonlinear variables." They occur nonlinearly in either the objective function or the first m_1 constraints. There may be purely linear constraints, given by (3.3). As before, a full set of slack variables is included as the last m components of the "linear variables" $\underline{y}$, so that general equality and inequality constraints can be accommodated in (3.2) and (3.3) by means of suitable bounds in (3.4).

We shall assume that the functions $f^i(\underline{x})$ are twice continuously differentiable with gradients $\underline{g}^i(\underline{x})$ and bounded Hessians $G^i(\underline{x})$, $i = 0,1,\ldots, m_1$. We shall also assume that the 1st and 2nd order Kuhn-Tucker conditions hold for a local minimum $\underline{x}^*$ with corresponding $\underline{\lambda}^*$.

The solution process consists of a sequence of "major iterations," each one involving a linearization of the nonlinear constraints at some point $\underline{x}_k$, corresponding to a first-order Taylor's series approximation:

$$f^i(\underline{x}) = f^i(\underline{x}_k) + \underline{g}^i(\underline{x}_k)^T(\underline{x} - \underline{x}_k) + O\|\underline{x} - \underline{x}_k\|^2 \quad .$$

We thus define

$$\underline{\tilde{f}}(\underline{x},\underline{x}_k) = \underline{f}(\underline{x}_k) + J(\underline{x}_k)(\underline{x} - \underline{x}_k) \ ,$$

or

$$\underline{\tilde{f}} = \underline{f}_k + J_k(\underline{x} - \underline{x}_k) \ , \tag{3.5}$$

where $J(\underline{x})$ is the $(m_1 \times n_1)$ Jacobian matrix whose ij-th element is $\partial f^i/\partial x_j$. We then see that

$$\underline{f} - \tilde{\underline{f}} = (\underline{f} - \underline{f}_k) - J_k(\underline{x} - \underline{x}_k) \tag{3.6}$$

consists of the higher order ("nonlinear") terms in the Taylor's expansion of $f(\underline{x})$ about the point $\underline{x}_k$.

3.2. The Linearized Subproblem

At the kth major iteration the following linearly constrained subproblem is solved:

$$\underset{\underline{x},\underline{y}}{\text{minimize}}\ L(\underline{x},\underline{y},\underline{x}_k,\underline{\lambda}_k,\rho) = f^0(\underline{x}) + \underline{c}^T\underline{x} + \underline{d}^T\underline{y} - \underline{\lambda}_k^T(\underline{f} - \tilde{\underline{f}}) + \frac{1}{2}\rho(\underline{f} - \tilde{\underline{f}})^T(\underline{f} - \tilde{\underline{f}}) \tag{3.7}$$

subject to

$$\tilde{\underline{f}} + A_1\underline{y} = \underline{b}_1 , \tag{3.8}$$

$$A_2\underline{x} + A_3\underline{y} = \underline{b}_2 , \tag{3.9}$$

$$\underline{\ell} \le \begin{bmatrix} \underline{x} \\ \underline{y} \end{bmatrix} \le \underline{u} . \tag{3.10}$$

The derivative of the objective function with respect to $\underline{x}$ is

$$\frac{\partial L}{\partial \underline{x}} = \underline{g}(\underline{x},\underline{x}_k,\underline{\lambda}_k,\rho) = \underline{g}^0(\underline{x}) + \underline{c} - (J-J_k)^T[\underline{\lambda}_k - \rho(\underline{f} - \tilde{\underline{f}})] . \tag{3.11}$$

We see that the nonlinearities in L involve $\underline{x}$ but not $\underline{y}$. (In contrast the normal penalty function would involve the constraint violation $\underline{f} + A_1\underline{y} - \underline{b}_1$ in place of $\underline{f} - \tilde{\underline{f}}$.) This represents a vital advantage of the modified augmented Lagrangian (3.7), since it means that each linearized subproblem has the same number of nonlinear variables as the original problem. The dimension of the reduced Hessian for the subproblem is therefore bounded in the same way as for the original problem, i.e., by the dimension of $\underline{x}$ (cf. Theorem 1 of [13]).

The use of a penalty term to ensure the augmented Lagrangian maintains a positive-definite Hessian in the appropriate subspace was first suggested by Arrow and Solow [2] and adopted later by, among others, Hestenes [10] and Powell [16] in their sequential unconstrained procedures, and by Sargent and Murtagh [23] in conjunction with their "variable-metric projection" algorithm involving a sequence of linearized constraints. The modified penalty term has not been used elsewhere since no distinction has been made previously between linear and nonlinear variables. Note that the modified penalty is identical to the conventional penalty in the subspace defined by the linearized constraints.

3.3. Choice of $\underline{\lambda}_k$

The choice $\underline{\lambda}_k = \underline{0}$, $\rho = 0$ corresponds to simple sequential linearization of the nonlinear constraints with no additional terms to $f^0(\underline{x})$ in the objective function. We shall call this the 'Newton strategy,' although it should not be confused with applying Newton's method to the Kuhn-Tucker equations for a solution of (3.1)-(3.4).

Ideally, $\underline{\lambda}_k$ should be as close as possible to $\underline{\lambda}^*$, but of course the optimal multipliers are normally unknown. The simplest choice is $\underline{\lambda}_k = \hat{\underline{\lambda}}$, the multipliers corresponding to linearized constraints at the solution of the previous subproblem. As we shall see, this choice is the best of several alternatives. For convenience suppose there are no linear constraints, so that $\hat{\underline{\lambda}} = \underline{\pi}$ is the solution of $B^T\underline{\pi} = g_B$ at the end of the previous major iteration. We know that $\underline{\pi}$ also satisfies $S^T\underline{\pi} = g_S$

(at least to within the convergence tolerance used for the subproblem). We thus have

$$\begin{bmatrix} B^T \\ \\ S^T \end{bmatrix} \underline{\hat{\lambda}} = \begin{bmatrix} \underline{g}_B \\ \\ \underline{g}_S \end{bmatrix}$$

and it is immaterial which variables are in B and which are in S. Now $\underline{g}$ is zero for all slack variables and it follows immediately that $\hat{\lambda}_i = 0$ if the ith linearized constraint is inactive. The choice $\underline{\lambda}_k = \underline{\hat{\lambda}}$ therefore ensures that an apparently inactive nonlinear constraint will be excluded from the Lagrangian term $\underline{\lambda}_k^T(\underline{f} - \underline{\tilde{f}})$ in the next subproblem. This is a desirable property.

It may seem that a better approximation to $\underline{\lambda}^*$ could be obtained by evaluating the new Jacobian $J(\underline{\hat{x}})$ which is required anyway for the next subproblem. Let the resulting "new" $[B \ \ S]$ be denoted by $[\bar{B} \ \ \bar{S}]$. One possibility is to define $\underline{\lambda}_k$ as the solution of the least-squares problem

$$\begin{bmatrix} \bar{B}^T \\ \bar{S}^T \end{bmatrix} \underline{\lambda} \approx \begin{bmatrix} \underline{g}_B \\ \underline{g}_S \end{bmatrix}$$

where the rhs is still the "old" gradient vector for the previous augmented Lagrangian. However, this least-squares problem would be very expensive to solve for large problems. Furthermore it is not guaranteed that $\lambda_i = 0$ would result where desired.

A cheaper alternative would be to solve $\bar{B}^T\underline{\bar{\pi}} = \underline{g}_B$ and take $\underline{\lambda}_k = \underline{\bar{\pi}}$, but then $\lambda_i = 0$ for inactive constraints would be assured only if the corresponding slack variable happened to be basic and not superbasic.

If the new $\bar{B}$ is to be used, the method of Sargent and Murtagh [23] shows that

$$\bar{B}^T \underline{\lambda} = \underline{g}_B + [I \quad 0 \quad 0] G_L \, \Delta \underline{x}$$

would produce the correct multipliers for the solution of the new subproblem if the original objective and constraints were quadratic and G_L was an adequate approximate to the Hessian of the new Lagrangian. (See equation (12) in [13].) However, this again is not a practical alternative for large problems.

3.4. Choice of ρ

It is well known that $\underline{x}^*$ need not be a local minimum of the Lagrangian function (with $\rho = 0$). If we assume that $J(\underline{x}^*)$ is of full rank, then $\underline{\lambda}^*$ exists and is such that

$$L(\underline{x},\underline{\lambda}) = f^0(\underline{x}) + \underline{c}^T \underline{x} + \underline{d}^T \underline{y} - \underline{\lambda}^T [\underline{f} + A_1 \underline{y} - \underline{b}_1]$$

is stationary at $(\underline{x}^*,\underline{\lambda}^*)$, but $L(\underline{x}^*,\underline{\lambda}^*)$ may well display negative curvature in $\underline{x}$ at $\underline{x}^*$.

The most that can be said [26] is that, if we consider the constraints satisfied at $\underline{x}^*$ as equalities and ignore the inactive ones, then a necessary (sufficient) condition that x^* is a local minimum is

$$Z(\underline{x}^*)^T \frac{\partial L}{\partial x} (\underline{x}^*,\underline{\lambda}^*) = \underline{0}$$

and

$$Z(\underline{x}^*)^T \frac{\partial^2 L}{\partial \underline{x}^2} (\underline{x}^*,\underline{\lambda}^*) Z(\underline{x}^*)$$

is positive semidefinite (positive definite), where $Z(\underline{x}^*)$ is as defined in equation (2.5) using $J(\underline{x}^*)$ in the appropriate part of A.

Thus if we restrict our search to the linearly constrained subspace defined by $Z(\underline{x}^*)$ we do indeed seek a minimum of the Lagrangian, and we may expect that when $\underline{x}_k$ is sufficiently close to $\underline{x}^*$ for $J(\underline{x}_k)$ to be close to $J(\underline{x}^*)$ we may minimize (3.7) with $\rho = 0$. This is confirmed by Robinson's theorem on quadratic convergence [20].

Difficulty arises when $\underline{x}_k$ is far removed from $\underline{x}^*$, since the linearized constraints may define a subspace where perhaps a saddle-point would be closer to $\underline{x}^*$ than a minimum would be. Successive minima of (3.7)-(3.10) with $\rho = 0$ may therefore fail to converge to $\underline{x}^*$. The addition of a penalty term $\rho[\underline{f}-\tilde{\underline{f}}]^T[\underline{f}-\tilde{\underline{f}}]$ imposes the correct curvature properties on (3.7) for a sufficiently large $\rho > 0$.

For general nonconvex problems it is not practical to determine a priori what the appropriate order of magnitude ρ should be (indeed $\rho = 0$ is often adequate even in the nonconvex case). The more important consideration is when to reduce ρ to zero, for we know that there is a radius of convergence around $(\underline{x}^*,\underline{\lambda}^*)$ within which Robinson's theorem holds for $\rho = 0$, and we can then expect a quadratic rate of convergence.

Two parameters we can monitor at the solution $\hat{\underline{x}}$ to each linearized subproblem are the constraint violation or 'row error',

$$\|\underline{f}(\hat{\underline{x}}) + A_1\underline{y} - \underline{b}_1\| = \|\underline{f}(\hat{\underline{x}}) - \tilde{\underline{f}}(\hat{\underline{x}},\underline{x}_k)\| ,$$

and the change in multiplier estimates, $\|\hat{\underline{\lambda}} - \underline{\lambda}_k\|$. The question that arises is whether these can be used to provide adequate measures of convergence toward $\underline{x}^*$.

For simplicity, consider the equality-constrained problem

$$P_0: \quad \text{minimize} \quad f^0(\underline{x})$$

$$\text{subject to} \quad \underline{f}(\underline{x}) = \underline{0}$$

where the functions of $\underline{x}$ are twice continually differentiable with bounded Hessians. We shall assume that at some point $\underline{x}^*$ the Jacobian $J(\underline{x}^*)$ is of full rank, there exists a $\underline{\lambda}^*$ such that $\partial f^0/\partial \underline{x} = J(\underline{x}^*)^T \underline{\lambda}^*$, and the reduced Hessian $Z(\underline{x}^*)^T \; \partial^2 L(\underline{x}^*, \underline{\lambda}^*)/\partial \underline{x}^2 \; Z(\underline{x}^*)$ is positive definite (i.e the sufficiency conditions are satisfied for $\underline{x}^*$ to be a local optimum).

Theorem 1.

Let $(\underline{x}_k, \underline{\lambda}_k)$ be an approximate solution to P_0 and let $(\underline{\hat{x}}, \underline{\hat{\lambda}})$ be a solution to the linearized subproblem

$$S_1: \quad \text{minimize} \quad f^0(\underline{x}) - \underline{\lambda}_k^T(\underline{f} - \underline{\tilde{f}}) + \frac{1}{2}\rho(\underline{f} - \underline{\tilde{f}})^T(\underline{f} - \underline{\tilde{f}})$$

$$\text{subject to} \quad \underline{\tilde{f}}(\underline{x}, \underline{x}_k) = \underline{0} \; .$$

If $\underline{\hat{\lambda}} - \underline{\lambda}_k = \underline{\varepsilon}_1$ and $\underline{f}(\underline{\hat{x}}) = \underline{\varepsilon}_2$, then $(\underline{\hat{x}}, \underline{\hat{\lambda}})$ is also a solution to the perturbed problem

$$P_1: \quad \text{minimize} \quad f^0(\underline{x}) + (\underline{\varepsilon}_1 + \rho\underline{\varepsilon}_2)^T (\underline{f} - \underline{\tilde{f}})$$

$$\text{subject to} \quad \underline{f}(\underline{x}) = \underline{\varepsilon}_2$$

for sufficiently small $\underline{\varepsilon}_1$ and $\underline{\varepsilon}_2$.

<u>Proof.</u> If $(\underline{\hat{x}}, \underline{\hat{\lambda}})$ is a solution of S_1 we must have $\underline{\tilde{f}} = \underline{0}$ and

$$\underline{g}^0(\underline{\hat{x}}) - (\hat{J} - J_k)^T \underline{\lambda}_k + \rho(\hat{J} - J_k)^T(\underline{\hat{f}} - \underline{\tilde{f}}) = J_k^T \underline{\hat{\lambda}}$$

where J_k is the Jacobian at $\underline{x}_k$ but $\hat{J}$, $\underline{\hat{f}}$ and $\underline{\tilde{f}}$ are evaluated at $\underline{\hat{x}}$. Adding $(\hat{J} - J_k)^T \underline{\hat{\lambda}}$ to both sides and inserting the expressions for $\underline{\varepsilon}_1$ and $\underline{\varepsilon}_2$ gives

$$\underline{g}^0(\underline{\hat{x}}) + (\hat{J} - J_k)^T \underline{\varepsilon}_1 + \rho(\hat{J} - J_k)^T \underline{\varepsilon}_2 = \hat{J}^T \underline{\hat{\lambda}}$$

which shows that $(\underline{\hat{x}}, \underline{\hat{\lambda}})$ also satisfies the conditions for a stationary point of P_1. Now it can be shown that the Hessians for the Lagrangian functions of S_1 and P_1 differ only by the amount $\rho(\hat{J} - J_k)^T(\hat{J} - J_k)$ at the solution of P_1, which is of order $\rho\|\Delta\underline{x}\|_k^2$ where $\Delta\underline{x}_k = \underline{\hat{x}} - \underline{x}_k$. Hence for sufficiently small $\underline{\varepsilon}_1$, $\underline{\varepsilon}_2$ and $\Delta\underline{x}_k$, if the reduced Hessian of S_1 is positive definite at $\underline{\hat{x}}$ then by continuity the reduced Hessian of P_1 will also be positive definite, thus satisfying the sufficiency conditions for a local minimum of P_1 at $\underline{\hat{x}}$. □

It is of interest to examine the corresponding result for the conventional penalty term.

<u>Theorem 2.</u> Let $(\underline{x}_k, \underline{\lambda}_k)$ be an approximate solution to P_0 and let $(\underline{\hat{x}}, \underline{\hat{\lambda}})$ be a solution to the linearized subproblem

S_2: minimize $f^0(\underline{x}) - \underline{\lambda}_k^T(\underline{f} - \underline{\tilde{f}}) + \frac{1}{2}\rho\underline{f}^T\underline{f}$

subject to $\underline{\tilde{f}}(\underline{x},\underline{x}_k) = \underline{0}$.

If $\hat{\underline{\lambda}} - \underline{\lambda}_k = \underline{\varepsilon}_1$ and $\underline{f}(\hat{\underline{x}}) = \underline{\varepsilon}_2$, then $(\hat{\underline{x}},\hat{\underline{\lambda}})$ is also a solution to the perturbed problem

P_2: minimize $f^0(\underline{x}) + \underline{\varepsilon}_1^T(\underline{f} - \underline{\tilde{f}}) + \rho\underline{\varepsilon}_2^T\underline{f}$

subject to $\underline{f}(\underline{x}) = \underline{\varepsilon}_2$.

Proof. Analogous to the proof of Theorem 1. □

Again it follows that if $\underline{\varepsilon}_1$ and $\underline{\varepsilon}_2$ are sufficiently small, $(\hat{\underline{x}},\hat{\underline{\lambda}})$ will be within the radius of convergence of Robinson's theorem and ρ can safely be reduced to zero. A point of interest is that problem P_1 appears to be less sensitive than P_2 to deviations from its optimum. Thus, let $\Delta\underline{x}$ be an arbitrary small change to the solution $\hat{\underline{x}}$ of P_1. The objective function for P_1 then differs from the true objective $f^0(\underline{x})$ by an amount

$$\delta_1 = (\underline{\varepsilon}_1 + \rho\underline{\varepsilon}_2)^T(\underline{f} - \underline{\tilde{f}}) ,$$

$$|\delta_1| \le (\|\underline{\varepsilon}_1\| + \rho\|\underline{\varepsilon}_2\|)\, O\|\Delta\underline{x}\|^2 .$$

For P_2 the analogous deviation is

$$\delta_2 = \underline{\varepsilon}_1^T(\underline{f} - \tilde{\underline{f}}) + \rho\underline{\varepsilon}_2^T\underline{f}$$

$$= \underline{\varepsilon}_1^T(\underline{f} - \tilde{\underline{f}}) + \rho\underline{\varepsilon}_2^T(\tilde{\underline{f}} + J\Delta\underline{x} + O\|\Delta\underline{x}\|^2) ,$$

$$|\delta_2| \leq (\|\underline{\varepsilon}_1\| + \rho\|\underline{\varepsilon}_2\|)\, O\|\Delta\underline{x}\|^2 + \rho\|\underline{\varepsilon}_2\|^2 + \rho\|J^T\underline{\varepsilon}_2\|\;\|\Delta\underline{x}\| .$$

Since δ_1 is of order $\|\Delta\underline{x}\|^2$ while δ_2 is of order $\|\Delta\underline{x}\|$, it appears that the modified penalty term in S_1 has a theoretical advantage over the conventional penalty term of S_2.

3.5. Summary of Procedure

The cycle of major iterations can be described as follows:

Step 0. Set $k = 0$. Choose some initial estimates $\underline{x}_0$, $\underline{\lambda}_0$ and specify a penalty parameter $\rho \geq 0$ and a convergence tolerance $\varepsilon_c > 0$.

Step 1. (a) Given $\underline{x}_k$, $\underline{\lambda}_k$ and ρ, solve the linearly constrained problem (3.7)-(3.10) to obtain new quantities $\underline{x}_{k+1}$, $\underline{y}_{k+1}$ and $\underline{g}(\underline{x}_{k+1})$.

(b) Solve $B^T\underline{\pi} = \underline{g}_B(\underline{x}_{k+1})$.

(c) Set $\underline{\lambda}_{k+1}$ = the first m_1 components of $\underline{\pi}$.

Step 2. (a) Test $\underline{x}_{k+1}$ for convergence. If optimal, exit.

(b) If $\|\underline{f}(\underline{x}_{k+1}) + A_1\underline{y}_{k+1} - \underline{b}_1\|/(1+\|(\underline{x}_{k+1},\underline{y}_{k+1})\|) \leq \varepsilon_c$ and $\|\underline{\lambda}_{k+1} - \underline{\lambda}_k\|/(1 + \|\underline{\lambda}_{k+1}\|) \leq \varepsilon_c$, then set $\rho = 0$.

(c) Relinearize the constraints at $\underline{x}_{k+1}$.

(d) Set $k = k+1$ and repeat from Step 1.

This procedure would not be complete without an algorithm for increasing the penalty parameter in certain circumstances. In Step 2(b) of the present implementation, we raise ρ by some factor if the relative change in $\underline{\lambda}_k$ proves to be very large.

4. COMPUTER IMPLEMENTATION

4.1. Sparse Matrices

Using equation (3.5), the linearized constraints (3.8) can be expressed in the form:

$$J_k \underline{x} + A_1 \underline{y} = \underline{b}_1 + J_k \underline{x}_k - \underline{f}_k \tag{4.1}$$

where $\underline{f}_k = \underline{f}(\underline{x}_k)$. The terms on the right-hand side of (4.1) are constant and become part of "$\underline{b}$", the current right-hand side. The set of linear constraints "$A\underline{x} = b$" for each major iteration is thus of the form:

$$\begin{bmatrix} J_k & A_1 \\ A_2 & A_3 \end{bmatrix} \begin{bmatrix} \underline{x} \\ \underline{y} \end{bmatrix} = \begin{bmatrix} \underline{b}_1 + J_k \underline{x}_k - \underline{f}_k \\ \underline{b}_2 \end{bmatrix} . \tag{4.2}$$

The major implication of A being large and sparse is that efficient methods are available for forming and updating an LU factorization of the basis matrix B (cf. equation (2.4)). (In particular, a "bump and spike" algorithm [9] is used to preserve sparsity at each refactorization of B. This occurs at the start of every relinearization and occasionally thereafter as necessary. At each intervening change of basis the LU factors are updated using the scheme described by Saunders [24] to preserve both sparseness and numerical stability.)

4.2. Infeasible Subproblems

One of the difficulties with sequential linearization is that some of the linearized subproblems may prove to be infeasible. In particular, the point $(\underline{x}_k, \underline{y}_k)$ used to define subproblem k is usually not a feasible point for the subproblem. However it will typically be feasible for the previous subproblem (and possibly optimal). This can be used to advantage in the manner suggested by Powell [18]. Thus we write the linearized constraints (4.1) in the form

$$J_k \underline{x} + A_1 \underline{y} = \underline{b}_1 + J_k \underline{x}_k - \underline{f}_k + \gamma \underline{q} \tag{4.3}$$

where $\gamma \underline{q}$ is a perturbation to the right-hand side. If $(\underline{x}_k, \underline{y}_k)$ is the final feasible point from subproblem $k - 1$, we can show that it will also be feasible with respect to the new linearized constraints (4.3) if $\gamma = 1$ and $\underline{q} = \underline{f}(\underline{x}_k) - \underline{\tilde{f}}(\underline{x}_k, \underline{x}_{k-1})$. (Thus $\underline{q}$ is the value of $\underline{f} - \underline{\tilde{f}}$ at the end of the previous major iteration.)

In MINOS/AUGMENTED the right-hand side of (4.3) is initialized with $\gamma = 0$. If the subproblem proves to be infeasible we add $\frac{1}{2}\underline{q}$ to the right-hand side and continue the solution process. If there is still no feasible solution we add $\frac{1}{4}\underline{q}$, $\frac{1}{8}\underline{q}$ and so on. This simulates the sequence of values $\gamma = \frac{1}{2}, \frac{3}{4}, \frac{7}{8}, \ldots$ tending to 1 as desired.

If the above procedure fails after 10 modifications, or if it is not applicable (e.g. when $k = 0$ or the previous subproblem was infeasible), a new linearization is requested as long as at least one minor iteration has been performed. Otherwise the algorithm is terminated with the assumption that the original problem itself is infeasible.

In [21], Rosen guards against infeasible subproblems by linearizing perhaps only some of the nonlinear constraints, namely those that have been active or reasonably close to active at any earlier stage. This alternative could be implemented in MINOS/AUGMENTED by adjusting the bounds on the slack variables associated with the linearized constraints.

4.3. User Options

Various implementation options are discussed in the following sections. Capitalized keywords at the head of each section illustrate the input data needed to select any particular option. Fuller details are given in the user's manual [14].

4.4. Subroutines CALCFG and CALCON

VERIFY OBJECTIVE GRADIENT

VERIFY CONSTRAINT GRADIENTS

As in the linearly constrained version of MINOS, a user-written subroutine CALCFG is required to calculate the objective function $f^0(\underline{x})$ and its gradient. The Lagrangian terms in (3.7) are calculated internally.

The user also supplies a subroutine CALCON to define the constraint vector $\underline{f}(\underline{x})$ and the current Jacobian $J(\underline{x})$. The nonzeros in J are returned column-wise in an output vector and must be in the same order as the corresponding entries in the MPS file (see below).

Subroutine CALCON is called every time the constraints are linearized. Except for Newton's method it is also called one or more times each linesearch to allow computation of (3.7) and (3.11). The expense of evaluating the constraints and their gradients should therefore be taken into account when specifying the linesearch accuracy.

Note that every function and Jacobian element is computed in every call to CALCON. Although some of these values may effectively be wasted (e.g. if some of the constraints are a long way from being active), the resulting simplicity of the subroutine from the user's point of view cannot be overemphasized.

Since the programming of gradients is notoriously prone to error, the VERIFY option is an essential aid to the user. This requests a check on the output from CALCFG and/or CALCON, using finite differences of $f^0(\underline{x})$ or $\underline{f}(\underline{x})$ along the coordinate directions. The check is performed at the first feasible point obtained (where feasibility is with respect to the first linearized subproblem). This point will not satisfy the nonlinear constraints in general, but at least it will satisfy the linear constraints and the upper and lower bounds on $\underline{x}$. Hence it is usually possible to avoid singularities in the nonlinear functions, both in the gradient check and in subsequent iterations.

4.5. Jacobian Option

JACOBIAN = SPARSE *or* DENSE

The submatrices A_1, A_2, A_3 and vectors $\underline{b}_1$, $\underline{b}_2$ in equation (4.2) are constant data and so may be entered using a standard MPS input file, as in linear programming, whereby only the nonzero coefficients and their row locations are entered column-by-column. Since we envisage that the Jacobian submatrix J will also be large and sparse we use the same scheme for recording the row and column locations of the nonzeros. Thus (with JACOBIAN = SPARSE) the sparsity pattern of J is entered as part of the MPS file. The corresponding numerical values in the MPS file may be genuine coefficients (if they are constant) or else dummy values, such as zero. Each call to subroutine CALCON subsequently replaces all dummy entries by their actual value at the current point $\underline{x}$.

Of course the intention here is to allow use of standard matrix generators to specify as much of the constraint matrix as possible. Pinpointing the nonzeros of J by name rather than number has the usual advantages, and in subroutine CALCON some code of the form

```
LJAC = LJAC + 1
G(LJAC) = ...
```

is usually adequate to define the next nonzero in a column of the Jacobian, without explicit reference to any row or column numbers. Nevertheless, the user is effectively required to give the sparsity pattern twice (in the MPS file and in CALCON), and it is essential that mismatches be avoided. At present the VERIFY option is the only aid to detecting incompatibility.

In the interest of simplicity, the option JACOBIAN = DENSE allows J to be treated as a dense matrix. In this case the MPS file need not specify any elements of J, and subroutine CALCON can use assignment statements of the form G(I,J) = ... to specify J_{ij} by row and column number. The danger of mismatches is thereby eliminated, but the storage requirements may be excessive for large problems. It may also give rise to an unnecessarily large "bump" in the basis factorizations.

4.6. Partial Completion

COMPLETION = PARTIAL or FULL

"Partial completion" is a compromise between the two extremes of relinearizing after each linesearch and solving each subproblem to high accuracy ("full completion").

The idea of attaining only partial completion at each major iteration can be accommodated effectively via the convergence tolerances. MINOS uses relatively loose tolerances for minimizing the reduced objective, until it appears that the optimal partition [B S N] has been identified. The partial completion option is effected by terminating a major iteration at this stage.

Otherwise for full completion the normal optimization procedure is continued using tighter tolerances to measure the change in $\underline{x}$, the size of the reduced gradient $(\| Z^T\underline{g}\|/\|\underline{\pi}\|)$, etc. This usually gives only small changes in $\underline{x}$ and $\underline{\pi}$ without changing the partition [B S N].

An alternative means for achieving partial completion for early major iterations is via the MINOR ITERATION limit (see Section 4.8).

4.7. Lagrangian Option, Penalty Parameter

Newton Strategy:	LAGRANGIAN	NO
	PENALTY PARAMETER	0.0

With this option the constraint functions and gradients are evaluated only once per major iteration.

Augmented Lagrangian:	LAGRANGIAN	YES
	PENALTY PARAMETER	$\rho \quad (\rho \geq 0)$

Here the constraints and Jacobian are evaluated as often as the objective. Evaluation of the augmented Lagrangian and its gradient with $\rho > 0$ is negligibly more expensive than with $\rho = 0$.

4.8. Convergence Conditions

MAJOR ITERATIONS	60	
MINOR ITERATIONS	40	
RADIUS OF CONVERGENCE	ε_c	$(\approx 10^{-2})$
ROW TOLERANCE	ε_r	$(\approx 10^{-6})$

Apart from terminating within each major iteration, we also need a terminating condition for the cycle of major iterations (Step 3, Section 3.5).

The point $(\underline{x}_k, \underline{y}_k)$ is assumed to be a solution to the nonlinearly constrained problem (3.1)-(3.4) if both the following conditions are satisfied:

1. $(\underline{x}_k, \underline{y}_k)$ satisfies the nonlinear constraints (3.2) to within a predefined error tolerance, i.e.

$$|f^i(\underline{x}_k) + \underline{e}_i^T A_1 \underline{y}_k - \underline{e}_i^T \underline{b}_1| \leq \varepsilon_r(1 + \|(\underline{x}_k, \underline{y}_k)\|), \qquad i = 1,\ldots,m_1 .$$

2. $(\underline{x}_k, \underline{y}_k)$ satisfies the first-order Kuhn-Tucker conditions for a solution to the linearized problem.

The tolerance parameter ε_r is specified by the user, and was set equal to 10^{-6} for most of the test problems described in the subsequent sections.

If the "partial completion" option is used, then once these two criteria are satisfied, a switch to "full completion" is invoked to obtain an accurate solution for the current subproblem, as well as for any possible further linearizations.

The error tolerance ε_c is used to define a radius of convergence about $(\underline{x}^*, \underline{\lambda}^*)$ within which Robinson's theorem is assumed to hold. If the row error defined above and the relative change in Lagrange multipliers between successive subproblems are both less than ε_c in magnitude then the penalty term is dropped (i.e. ρ is set to 0.0).

The MINOR ITERATION limit is used to terminate each linearized subproblem when the number of linesearches becomes excessive. The limit of 40 was used in all the numerical experiments. A much small number would result in more frequent use of expensive housekeeping operations such as the refactorization of B in Step 3, Section 3.5. Similarly a much larger number may be wasteful; if significant changes to $\underline{x}$ have occurred then a new linearization is appropriate, while if there has been little progress then updating the Lagrangian information gives some hope of more rapid progress.

It must be noted that for some choices of $\underline{x}_0$, $\underline{\lambda}_0$ and ρ the sequence of major iterations may not converge. The MAJOR ITERATION limit provides a safeguard for such circumstances.

For a discussion of linearly constrained Lagrangian methods and their drawbacks see Wright [26], pp. 137-147.

In the present implementation of MINOS/AUGMENTED the only recourse would be to restart with a different initial $\underline{x}$ or a larger value for the penalty parameter ρ (or both).

5. TEST PROBLEMS

Most of the text examples reported here appear in the published literature. The last two examples are new and are described in detail. They can be made arbitrarily large by increasing one parameter.

5.1. Colville No. 2

This well known problem is one of the more testing of the Colville series of problems [3] and has been used frequently to compare different algorithms [1], [8], [20], [23]. It has a cubic objective function and 15 quadratic constraints. Even in this small problem the variables can be partitioned into linear and nonlinear sets, of dimension 10 and 5 respectively.

a) Feasible starting point.

b) Infeasible starting point.

5.2. Colville No. 3

This problem has a quadratic objective function of five variables and three quadratic constraints. It also has upper and lower bounds on all the variables, and upper and lower limits on the constraints. These can be accommodated effectively by using the BOUNDS and RANGES options in the MPS file; the BOUNDS option allows variables to be nonbasic at their upper or lower bound, and the RANGES option assigns both upper and lower bounds to the slack variables associated with the constraints (thus allowing the right-hand side to range between bounds).

a) Feasible starting point.

b) Infeasible starting point.

5.3. Colville No. 8

This is a typical process design problem, where some of the variables are determined by solving nonlinear equations. It has 3 independent variables and 7 constraints.

5.4. Powell's Problem [17]

This has 5 variables and 3 constraints. Although small, this is a good test problem as the nonlinearities in the constraints are quite significant.

$$\begin{aligned} \text{minimize} \quad & \exp(x_1 x_2 x_3 x_4 x_5) \\ \text{subject to} \quad & x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 = 10 , \\ & x_2 x_3 - 5 x_4 x_5 = 0 , \\ & x_1^3 + x_2^3 = -1 . \end{aligned}$$

Starting point: (-2, 2, 2, -1, -1).

5.5. Power Scheduling

This is a comparatively large test problem reported recently by Biggs and Laughton [4], with 79 variables and 91 constraints (all nonlinear). It also has upper and lower bounds on some of the variables. Although all the variables and constraints are nonlinear, the linearized constraint matrix J_k (equation 4.3) is quite sparse with on average a little under 6 nonzero row entries per column. Treating it as a dense matrix could result in a "bump" of 79 columns, which is clearly undesirable. A number of minor discrepancies between Biggs and Laughton's paper and the original statement of the problem [25] were resolved by using the original data.

5.6. Launch Vehicle Design

This problem appears in Bracken and McCormick's book on nonlinear programming applications [5] and also appears in [22]. There are 12 linear constraints and 10 nonlinear constraints, and the Jacobian of the nonlinear constraints is quite sparse (density 23%), yielding an overall matrix density of 15%. All 25 variables are nonlinear.

5.7. Quartic with Quartic Constraints

This problem appears in Pierre and Lowe [15]. Only a few terms are quartic, the remainder being predominantly quadratic. It has 20 variables (all nonlinear) and 17 constraints, 13 of which are nonlinear.

5.8. Dembo No. 7

This is one of Dembo's set of 8 Geometric Programming test problems [28]; in particular, it required the most computation time in Dembo's results. Other authors have reported difficulty with the problem (Powell [29], Coope and Fletcher [27]). There are 16 variables (all nonlinear) and 19 general constraints (11 of them nonlinear). The solution has a few primal and dual degeneracies, but it is essentially at a vertex of the constraint space (i.e., a vertex of the final linearization).

5.9. Wright No. 4 [26]

This is a highly nonlinear non-convex problem for which four local minima have been determined.

$$\text{minimize} \quad (x_1-1)^2 + (x_1-x_2)^2 + (x_2-x_3)^3 + (x_3-x_4)^4 + (x_4-x_5)^4$$

$$\text{subject to} \quad x_1 + x_2^2 + x_3^3 = 2 + 3\sqrt{2} ,$$

$$x_2 - x_3^2 + x_4 = -2 + 2\sqrt{2} ,$$

$$x_1 x_5 = 2 .$$

Starting points:

(a) (1,1,1,1,1)

(b) (2,2,2,2,2)

(c) (-1,3,-1/2,-2,-3)

(d) (-1,2,1,-2,-2)

(e) (-2,-2,-2,-2,-2)

Local optima:

$$\underline{x}^*(1) = (1.11663,\ 1.22044,\ 1.53779,\ 1.97277,\ 1.79110)^T$$

$$\underline{x}^*(2) = (-2.79087,\ -3.00414,\ .205371,\ 3.87474,\ -.716623)^T$$

$$\underline{x}^*(3) = (-1.27305,\ 2.41035,\ 1.19486,\ -.154239,\ -1.57103)^T$$

$$\underline{x}^*(4) = (-.703393,\ 2.63570,\ -.0963618,\ -1.79799,\ -2.84336)^T$$

5.10. Wright No. 9 [26]

This is another highly nonlinear example.

$$\text{minimize} \quad 10x_1x_4 - 6x_3x_2^2 + x_2x_1^3 + 9\sin(x_5 - x_3) + x_5^4x_4^2x_2^3$$

$$\text{subject to} \quad x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 \le 20\ ,$$

$$x_1^2x_3 + x_4x_5 \ge -2\ ,$$

$$x_2^2x_4 + 10x_1x_5 \ge 5\ .$$

Starting points:

(a) (1,1,1,1,1)

(b) (1.091, -3.174, 1.214, -1.614, 2.134)

Local optima:

$$\underline{x}^*(1) = (-.0814522,\ 3.69238,\ 2.48741,\ .377134,\ .173983)^T$$

$$\underline{x}^*(2) = (1.47963,\ -2.63661,\ 1.05468,\ -1.61151,\ 2.67388)^T$$

With the barrier trajectory algorithm, Wright [26] obtained convergence to $\underline{x}^*(1)$ from (a) and convergence to $\underline{x}^*(2)$ from (b). Note that starting point (b) was originally chosen to cause difficulty for the barrier algorithm and other methods that retain feasibility throughout.

5.11. Optimal Control

This problem investigates the optimal control of a spring, mass and damper system. It is adapted from [19]. While it is acknowledged that there may be simpler ways of solving the problem by taking specific advantage of the nature of the constraints, it serves the present purpose of providing a large, sparse test problem.

$$\text{minimize} \quad f(\underline{x},\underline{y},\underline{u}) = \frac{1}{2}\sum_{t=0}^{T} x_t^2$$

$$\text{subject to} \quad \left.\begin{aligned} x_{t+1} &= x_t + 0.2y_t \\ y_{t+1} &= y_t - 0.01y_t^2 - 0.004x_t + 0.2u_t \\ -0.2 &\le u_t \le 0.2 \\ y_t &\ge -1.0 \end{aligned}\right\} \quad t = 0,\ldots, T-1,$$

$$x_0 = 10, \quad y_0 = 0, \quad y_T = 0 \ .$$

Starting point: $x_t = 0$, $y_t = -1$, $t = 1,\ldots, T$.

For the results below we set $T = 100$. There are thus 202 nonlinear variables (x_t, y_t, $t = 0,\ldots, 100$) and 100 linear variables (u_t, $t = 0,\ldots, 99$). There are also 100 quadratic constraints and 100 linear constraints. Note that the nonlinear constraints are very sparse, with only 4 nonzeros per row.

The solution is dominated to a large extent by the lower bound on y_t; the optimal y_t decreases with increasing t and remains at -1.0 from $t = 20$ to $t = 40$, and then increases again, goes slightly positive and settles to 0.0 at $t = 100$. The corresponding values of x_t can be calculated directly from the linear constraint $x_{t+1} = x_t + 0.2y_t$. The optimal value of the objective is $\|\underline{x}\|^2/2 = 1186.382$.

5.12. Economic Growth Model (Manne [11])

This is a model for optimizing aggregate consumption over time. The variables are C_t, I_t and K_t which represent consumption, investment and capital in time period t for $t = 1, \ldots, T$.

Utility function: $$\text{maximize} \sum_{t=1}^{T} \beta_t \log_t C_t$$

Nonlinear constraints: $$\alpha_t K_t^b \geq C_t + I_t\,, \qquad 1 \leq t \leq T$$

Linear constraints: $$K_{t+1} \leq K_t + I_t\,, \qquad 1 \leq t < T$$

$$gK_T \leq I_T$$

Bounds: $$K_1 = I_0 + K_0$$

$$\left.\begin{array}{l} K_t \geq I_0 + K_0 \\ C_t \geq C_0 \\ I_t \geq I_0 \\ I_t \leq (1.04)^t I_0 \end{array}\right\} \quad 1 \leq t \leq T$$

Data: $\beta = 0.95, \quad b = 0.25, \quad g = 0.03,$

$C_0 = 0.95, \quad I_0 = 0.05, \quad K_0 = 3.0,$

$$\beta_t = \beta^t \quad \text{except} \quad \beta_T = \beta^T/(1-\beta) \ ,$$

$$\alpha_t = \alpha(1+g)^{(1-b)t} \quad \text{where} \quad \alpha = (C_0 + I_0)/K_0^b \ .$$

The objective function (regarded as a minimand) and the nonlinear constraints are both convex and separable. The same is true of the preceding optimal control problem. These examples should therefore be useful test problems for more specialized convex programming algorithms.

For test purposes we have used $T = 100$, which gives a problem with 200 general constraints and 300 variables. The optimal value of the utility function is 9.287547. All general constraints are active at the solution, and the first 74 upper bounds on I_t are also active.

It should be mentioned that specialized methods are known to economists for solving this problem, with and without the upper bounds ("absorptive capacities") on the variables I_t. However in more practical circumstances the model would be imbedded in a much larger model for which an analytical solution is not known. If the larger model contains no additional nonlinearities, the performance of MINOS/AUGMENTED should degrade only in proportion to the problem size.

6. RESULTS AND DISCUSSION

MINOS/AUGMENTED is implemented in standard Fortran. Various options can be selected at run-time by means of a SPECS file, and an initial point $\underline{x}_0$ can be specified by an INITIAL BOUNDS set in the file containing the constraint data. The latter is defined in the well-known MPS format used by commercial mathematical programming systems.

The following parameter values were used throughout:

LINESEARCH PARAMETER ETA	= 0.1	(moderately accurate search)
RADIUS OF CONVERGENCE	= 0.01	(ϵ_c in Section 3.5)
ROW TOLERANCE	= 10^{-6}	(ϵ_r in Section 4.7)
MINOR ITERATIONS LIMIT	= 40	(not active on small examples)

Run-times are reported below in order to allow comparison among various algorithmic options.

6.1. Problems 5.1-5.8

These examples were solved on a CDC Cyber 70. In all cases convergence was obtained with zero penalty parameter ρ. The results are summarized in Table 6.1. In general the partial completion option converged more rapidly than full completion. However example 5.4 illustrates that fewer major iterations are likely if subproblems are solved accurately once the correct subspace has been identified. This was observed in several other cases and is probably explained by the discussion of $\underline{\lambda}_k$ in Section 3.3. In terms of total run-time the Newton strategy was often superior, but it failed to converge on problems 5.4 and 5.5. This deficien

becomes more prominent in the examples below.

Problem 5.8 was run with two different minor-iteration limits (3 and 40, which could have been 3 and 15 without altering the comparison). The results illustrate that terminating major iterations early can sometimes help rather than hinder.

6.2. Problems 5.9-5.12

The results for these examples were obtained on an IBM 370/168 using the Fortran H Extended compiler with full optimization (OPT = 3). Computation was performed in double precision, but the constraint data were stored in single precision, including J_k in the linearization of $\underline{f}(\underline{x})$. This limits the achievable constraint error to about 10^{-6}, but that is usually adequate in practice. Full completion was used throughout.

Problem 5.9 (Wright No. 4)

This highly nonlinear problem illustrates the difficulties discussed in Section 3.4 when no penalty term is used. The Newton strategy and the Lagrangian algorithm with $\rho = 0$ both gave rise to subproblems which changed radically each major iteration.

The results using the Lagrangian algorithm with $\rho = 10$ and $\rho = 100$ are shown in Table 6.2.

Infeasible subproblems were encountered with starting point (e) using the penalty parameter $\rho = 10$, but the procedure discussed in Section 4.6 successfully overcame this difficulty. Case (e) was also the only one for which the larger ρ was important in stabilizing progress from the starting point to the solution.

Problem	5.1(a)				5.1(b)				5.2(a)		5.2(b)		5.3	
Method	N	L	N	L	N	L	N	L	N	L	N	L	N	L
Completion	P.C.		F.C.		P.C.		F.C.		P.C.		F.C.		P.C.	
Major itns	10	4	10	4	4	4	9	4	3	3	3	3	3	3
Total itns	31	41	58	43	32	27	59	38	5	5	4	4	11	10
Functions	52	65	93	77	54	42	86	55	7	7	6	6	31	54
Time (secs)	3.18	3.58	4.43	3.52	4.38	3.07	4.15	3.07	0.91	0.97	0.91	0.91	1.36	2.05

Problem	5.4			5.5		5.6		5.7	
Method	N	L	L	N	L	N	L	N	L
Completion	F.C.	F.C.	P.C.	P.C.		P.C.		P.C.	
Major itns		4	6		5	3	3	4	27
Total itns	*	18	10	*	100	26	26	41	91
Functions		26	17		69	60	60	73	147
Time (secs)		1.53	1.53		38.9	6.65	8.13	5.61	20.1

5.8, Method L, F.C.		
Minor itns limit	3	40
Major itns	6	7
Total itns	30	55
Functions	19	66
Time (secs)	2.27	3.56

Table 6.1. Summary of Results for Test Problems 5.1-5.8.

N Newton strategy ($\lambda_k = 0$, $\rho = 0$)

L Lagrangian algorithm, $\rho = 0$

P.C. Partial Completion

F.C. Full Completion

* Non-convergence

Starting Point	(a)		(b)		(c)		(d)		(e)	
ρ	100	10	100	10	100	10	100	10	100	10
Major itns	9	7	9	6	6	6	5	5	7	12
Total itns	47	33	41	25	24	26	22	21	30	67
Functions	84	53	70	40	46	42	37	30	55	120
Solution	$\underline{x}^*(1)$	$\underline{x}^*(1)$	$\underline{x}^*(1)$	$\underline{x}^*(1)$	$\underline{x}^*(4)$	$\underline{x}^*(4)$	$\underline{x}^*(3)$	$\underline{x}^*(3)$	$\underline{x}^*(2)$	$\underline{x}^*(3)$

Table 6.2. Results for Test Problem 5.9.

6.3. Problem 5.10 (Wright No. 9)

Again the Newton strategy and the Lagrangian algorithm with $\rho = 0$ failed to converge. Results for the Lagrangian algorithm with $\rho = 10$ and $\rho = 100$ are shown in Table 6.3.

Starting point	(a)		(b)	
ρ	100	10	100	10
Major itns	12	9	5	19
Total itns	92	71	32	201
Functions	183	146	61	386
Solution	$\underline{x}^*(1)$	$\underline{x}^*(1)$	$\underline{x}^*(2)$	$\underline{x}^*(3)$

Table 6.3. Results for Test Problem 5.10

A value of $\rho = 10$ is almost too low for starting point (b), the subproblem solutions changing radically as they did for $\rho = 0$, but finally converging to a new local optimum, $\underline{x}^*(3) = (-.07427, -3.69170, 2.48792, 0.37693, 0.18446)^T$.

In general the results for these last two problems are inferior to those obtained by Murray and Wright [12], [26] with their trajectory algorithms, in terms of total function evaluations. However, the difference is less than a factor of 4 in all cases, and averaged 2.2 over the seven starting points.

6.4. Problem 5.11 (Optimal Control)

Despite the large size of this problem both the Newton strategy and the Lagrangian algorithm with $\rho = 0$ converged rapidly.

Method	N	L ($\rho = 0$)
Major itns	6	6
Total itns	254	247
Functions	213	203
Time (secs)	10.55	11.56

Table 6.4. Results for Test Problem 5.11

Recall that procedure N evaluates the constraint functions only once per major iteration (in this case 6 times compared to 203 times for procedure L). If $\underline{f}(\underline{x})$ were more costly to compute, the time advantage would be that much greater. The Lagrangian algorithm displayed insensitivity to nonzero values of ρ in the range 0.01-10.0, taking the same iterations and calculations as shown in Table 6.4.

6.5. Problem 5.12 (Economic Growth)

Although this is also a convex problem the Newton strategy led to oscillation and no convergence. The Lagrangian algorithm did converge rapidly with $\rho = 0$. Without the upper bounds $I_t \leq (1.04)^t I_0$ it required 11 major iterations, 355 minor iterations, 859 function calculations and 34.3 seconds, and there were 99 superbasic variables at the optimal solution. However when these bounds were imposed, the optimal number of superbasics was only 25 and convergence was obtained in 7 major iterations, 183 minor iterations, 497 function calculations and 11.9 seconds. This illustrates the gains that are made when the presence of constraints reduces the dimensionality of the optimal subspace.

As an experiment on the effect of ρ on the rate of convergence the problem was solved several times with different values of ρ in the range $10^{-4} \leq \rho \leq 1.0$. (The tolerance ε_c for dropping ρ was set to zero, thus forcing ρ to stay constant for each run.) The results are shown in Figure 1. This is a plot of minor iterations versus log of the constraint violation or "row error," $\log_{10} \| \underline{f}(\underline{x}) + A_1 \underline{y} - \underline{b}_1 \|_\infty$, immediately following relinearization. The dots represent the end of each major iteration. Initially these occur every 40 iterations (the MINOR ITERATIONS limit) but later subproblems were solved accurately before the limit was reached. In fact the number of minor iterations reduces rapidly to only one or two per subproblem as convergence is approached. This behavior was also observed in all of the preceding examples.

It can be seen that higher values of ρ give lower row errors at the end of the first major iteration (as we would expect), but they lead to consistently slower convergence. It is interesting to note that rapid

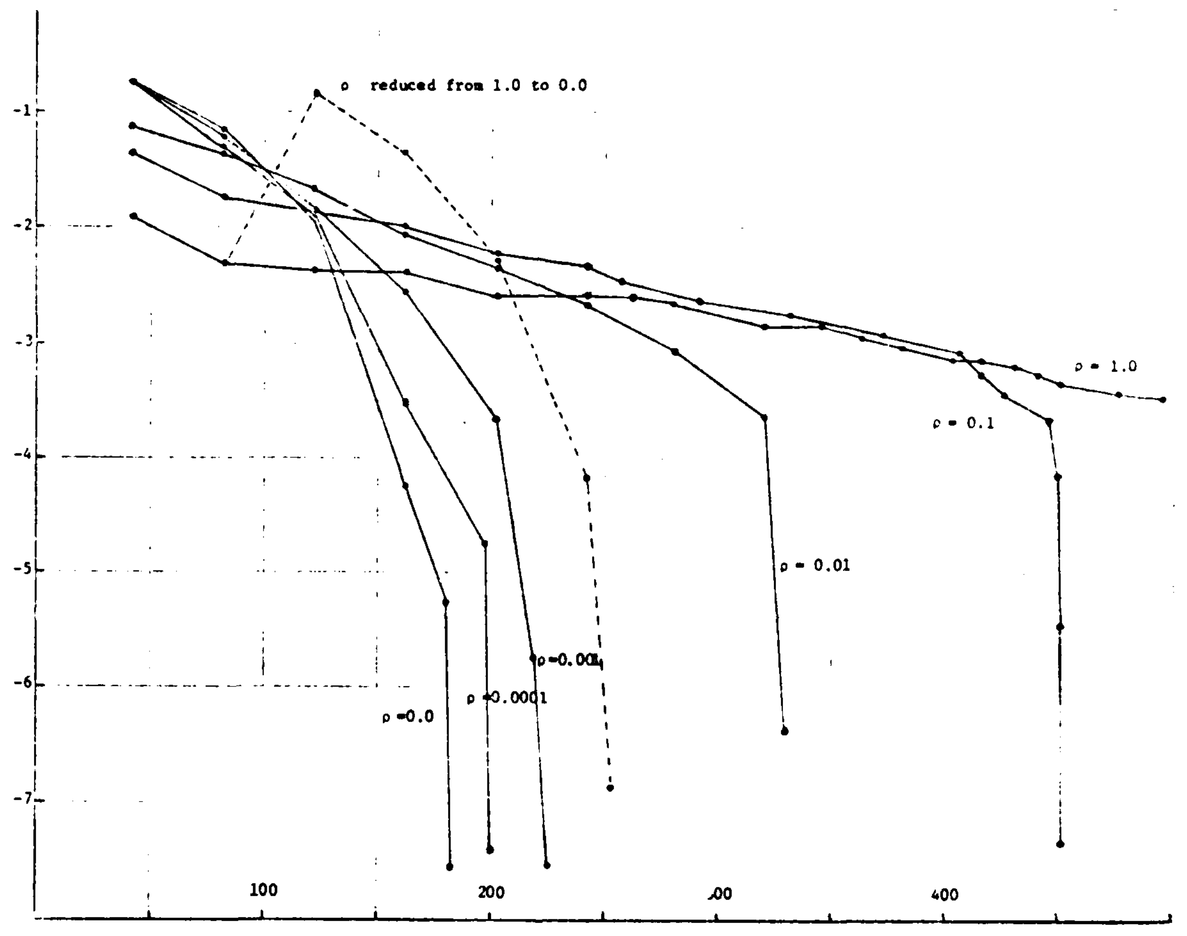

Figure 1. Solution of problem 5.12 using various values for the penalty parameter ρ. The constraint violation, $\log_{10}\|\underline{f}(\underline{x}) + A_1\underline{y} - \underline{b}_1\|_\infty$, is plotted against minor iteration number. Dots signify the end of each major iteration.

convergence does occur ultimately in all cases (except for $\rho = 1.0$ which was terminated after 500 iterations). However this is not until the correct active constraints have been identified, by which time $\underline{x}_k$ is very close to the optimum and the penalty term is having a negligible effect on the Lagrangian and its reduced gradient.

These results confirm that the benefit of Robinson's proof of quadratic convergence for the case $\rho = 0$ cannot be realized unless ρ is actually reduced to zero as soon as precautions allow.

The dotted line in Figure 1 shows the result for $\rho = 1.0$ (the worst case) with ε_c set to 0.01, allowing ρ to be reset to zero at the start of major iteration 3. (Note that on the highly nonlinear examples 5.9 and 5.10, the same value of ε_c ensured that ρ was not set to zero until a near optimum point was reached. This was the desired effect since the multiplier estimates $\underline{\lambda}_k$ were changing radically in the early major iterations.)

It will be seen in Figure 1 that the row error increases sharply once ρ is reset to zero. This is consistent with the algorithm suddenly being free to take a large step. We could therefore regard the first two major iterations as having served to verify that the problem is only mildly nonlinear.

7. CONCLUSIONS

Many real-life optimization problems originate as linear programming models that are not quite linear; i.e. they contain simple, differentiable functions in the constraints and objective, but otherwise the constraint set is large, sparse and linear. For such problems the Jacobian matrix is also likely to be sparse, and the strategy of solving a sequence of linearly constrained subproblems has many advantages. This is clear from the results obtained for the larger test problems above.

For convex problems the Lagrangian term in the objective of the subproblems is usually necessary to ensure convergence. The Newton strategy performed adequately without it on some occasions, but in general the saving in run time will seldom be substantial.

For non-convex problems, both the Lagrangian and the penalty term are clearly useful but the actual choice of the penalty parameter ρ remains a critical decision for the user. In practice, optimization problems are often solved repeatedly on a production basis. In this situation it is worthwhile experimenting with different values of the parameters and tolerances (and perhaps with the Newton strategy). However the case of an isolated problem with unknown characteristics is no less important. A conservative (high) value of ρ is then virtually mandatory. One of our aims has been to minimize the risk of subsequent slow convergence by determining an opportune time to reduce ρ to zero. The analysis in Section 3.4 has suggested a practical procedure for achieving this aim. In particular, the "radius of convergence" tolerance ε_c (applied to both the constraint violation and the relative change in the estimates of $\underline{\lambda}$) allows early removal of ρ in moderately nonlinear cases but otherwise retains it until convergence to a local solution is imminent.

The results reported here should provide a benchmark for measuring the performance of other algorithms and their implementations. Clearly no single algorithm can be expected to perform uniformly better than all others in such a diverse field as nonlinearly constrained optimization. As it happens, MINOS/AUGMENTED has proved to be reasonably efficient on small, highly nonlinear problems, but more importantly, it represents an advance in the development of general-purpose software for large-scale optimization.

ACKNOWLEDGMENTS

The authors wish to thank Noel Diaz, Dr. Philip Gill, Professor Alan Manne, Professor Walter Murray, Professor J. Ben Rosen, Ken Schofield, Wesley Winkler, and Dr. Margaret Wright for various guiding comments and assistance with the test problems.

REFERENCES

[1] Abadie, J. and J. Guigou, "Numerical Experiments with the GRG Method," in: J. Abadie, (ed.), Integer and Nonlinear Programming, North-Holland, Amsterdam, 1970, pp. 529-536.

[2] Arrow, K. J. and R. M. Solow in Studies in Linear and Nonlinear Programming, K. J. Arrow, L. Hurwicz and H. Uzawa (editors), Stanford University Press, Stanford, Calif. (1958), pp. 166-169.

[3] Colville, A. R., "A Comparative Study on Nonlinear Programming Codes," IBM New York Scientific Center Report 320-2949 (1968).

[4] Biggs, M. C. and M. A. Laughton, "Optimal Electric Power Scheduling: a Large Nonlinear Test Problem Solved by Recursive Quadratic Programming," Math. Programming 13 (1977), pp. 167-182.

[5] Bracken, J. and G. P. McCormick, Selected Applications of Nonlinear Programming, John Wiley, New York (1968), pp. 58-82.

[6] Gill, P. E. and W. Murray, "Newton-type Methods for Unconstrained and Linearly Constrained Optimization," Math. Programming 7 (1974), pp. 311-350.

[7] Gill, P. E. and W. Murray, "Safeguarded Steplength Algorithms for Optimization Using Descent Methods," Report NAC 37 (1974), National Physical Laboratory, Teddington.

[8] Garcia-Palomares, U. M. and O. L. Mangasarian, "Superlinearly convergent quasi-Newton Algorithms for Nonlinearly Constrained Optimization Problems," Math. Programming 11 (1976), pp. 1-13.

[9] Hellerman, E. and D. C. Rarick, "The Partitioned Preassigned Pivot Procedure," in: D. J. Rose and R. A. Willoughby (eds.), Sparse Matrices and Their Applications, Plenum Press, New York (1972), pp. 67-76.

[10] Hestenes, M. R., "Multiplier and Gradient Methods," J. Opt. Theory and Appl. 4 (1969), pp. 303-320.

[11] Manne, A. S., Private Communication (1979).

[12] Murray, W. and M. H. Wright, "Projected Lagrangian Methods Based on the Trajectories of Penalty and Barrier Functions," Report SOL 78-23 (1978), Dept. of Operations Research, Stanford University, California.

[13] Murtagh, B. A. and M. A. Saunders, "Large Scale Linearly Constrained Optimization," Math. Programming 14 (1978), pp. 41-72.

[14] Murtagh, B. A. and M. A. Saunders, "MINOS/AUGMENTED Supplementary User's Manual," SOL Report, to appear (1980).

[15] Pierre, D. A. and M. J. Lowe, Mathematical Programming via Augmented Lagrangians, Addison-Wesley, Reading (1975), pp. 239-241.

[16] Powell, M. J. D., "A Method for Nonlinear Constraints in Optimization Problems," in Optimization, R. Fletcher (ed.), Academic Press, New York, (1969), pp. 283-297.

[17] Powell, M. J. D., "Algorithms for Nonlinear Constraints that Use Lagrangian Functions," Math. Programming 14 (1978), pp. 224-248.

[18] Powell, M. J. D., "A Fast Algorithm for Nonlinearly Constrained Optimization Calculations," paper presented at 1977 Dundee Conference on Numerical Analysis.

[19] Ritch, P. S., "Discrete Optimal Control with Multiple Constraints U: Constraint Separation and Transformation Technique," Automatica 9 (1973), pp. 415-429.

[20] Robinson, S. M., "A Quadratically Convergent Algorithm for General Programming Problems," Math. Programming 3 (1972), pp. 145-156.

[21] Rosen, J. B., "Two-phase Algorithm for Nonlinear Constraint Problems," Tech. Rept. 77-22,(1977), Department of Operations Research, Stanford University, California.

[22] Rush, B. C., Bracken, J. and G. P. McCormick, "A Nonlinear Programming Model for Launch Vehicle Design and Costing," Operations Research 15 (1967), pp. 185-210.

[23] Sargent, R. W. H. and B. A. Murtagh, "Projection Methods for Nonlinear Programming," Math. Programming 4 (1973), pp. 245-268.

[24] Saunders, M. A., "A Fast, Stable Implementation of the Simplex Method using Bartels-Golub Updating," in: J. R. Bunch and D. J. Rose (eds.), Sparse Matrix Computations, Academic Press, New York (1976), pp. 213-226.

[25] Shen, C. M. and M. A. Laughton, "Determination of Optimum Power System Operating Conditions Under Constraints," Proceedings of the IEE 116 (1969), pp. 225-239.

[26] Wright, M. H., "Numerical Methods for Nonlinearly Constrained Optimization," SLAC Report No. 193 (1976), Stanford University, California (Ph.D. Dissertation).

[27] Coope, I. D. and R. Fletcher, "Some Numerical Experience with a Globally Convergent Algorithm for Nonlinearly Constrained Optimization," Report NA/30 (1979), Dept. of Mathematics, University of Dundee, Scotland.

[28] Dembo, R.S., "A Set of Geometric Programming Test Problems and their Solutions," Math. Programming 10 (1976), pp. 192-213.

[29] Powell, M.J.D., "Constrained Optimization by a Variable Metric Method," Report 77/NA6 (1977), Dept. of Applied Math. and Theoretical Physics, Cambridge University, England.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER SOL 80-1 / 2. GOVT ACCESSION NO. AD-A084 815	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) THE IMPLEMENTATION OF A LAGRANGIAN-BASED ALGORITHM FOR SPARSE NONLINEAR CONSTRAINTS.	5. TYPE OF REPORT & PERIOD COVERED TECHNICAL REPORT.
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) B.A. Murtagh [illegible] M.A. Saunders	8. CONTRACT OR GRANT NUMBER(s) DAAG-29-79-C-0110 DE-[illegible]3-76SF[illegible]
9. PERFORMING ORGANIZATION NAME AND ADDRESS Operations Research Department/SOL Stanford University Stanford, CA 94305	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS U.S. Army Research Office Box CM, Duke Station Durham, NC 20776	12. REPORT DATE January 1980
	13. NUMBER OF PAGES 48
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) [illegible]	15. SECURITY CLASS. (of this report) UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

This document has been approved for public release and sale; its distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

[illegible]

18. SUPPLEMENTARY NOTES

(11) 1[illegible]7[illegible].1-M

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

AUGMENTED LAGRANGIAN	NONLINEAR CONSTRAINTS	REDUCED-GRADIENT METHOD
LARGE-SCALE OPTIMIZATION	NONLINEAR PROGRAMMING	SIMPLEX METHOD
LINEAR CONSTRAINTS	OPTIMIZATION	SPARSE MATRIX
LINEAR PROGRAMMING	QUASI-NEWTON METHOD	VARIABLE-METRIC METHOD

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

SEE ATTACHED

SOL 80-1 by B.A. Murtagh and M.A. Saunders

THE IMPLEMENTATION OF A LAGRANGIAN-BASED ALGORITHM FOR SPARSE NONLINEAR CONSTRAINTS

An algorithm is described for solving large-scale nonlinear programs whose objective and constraint functions are smooth and continuously differentiable. The algorithm is of the augmented Lagrangian type, involving a sequence of sparse, linearly constrained subproblems whose objective functions include a modified Lagrangian term and a modified penalty function.

The algorithm has been implemented in a general purpose Fortran code called MINOS/AUGMENTED. The system is intended for use on problems whose Jacobian matrix is sparse. (Such problems usually include a large set of purely linear constraints.) The bulk of the data may be assembled using a standard linear-programming matrix generator. Function and gradient values for all nonlinear terms are supplied by two user-written subroutines.

Some aspects of the implementation are described in detail, and computational results are given for some nontrivial test problems. Assuming convergence occurs, the work involved is comparable to the solution of a moderate number of linear programs of similar size.